REMARKS

Claims 3-4, 6-19, and 24-28 were pending and rejected in a Final Office Action dated February 20, 2009. No claims are amended herein. In view of the Remarks that follow, Applicant respectfully requests that Examiner reconsider the outstanding claim rejections discussed below, and withdraw them.

Response to Rejection Under 35 USC § 102

In the Office Action, the Examiner rejected claim 27 under 35 USC § 102(e) as being anticipated by U.S. Patent No. 7,266,844 ("Teblyashkin"). This rejection is respectfully traversed.

Claim 27 is dependent on claim 6 and therefore includes all the limitations of claim 6. Specifically, claim 27 includes "identifying computer code ... having a decryption loop and a body", "optimizing the decryption loop to produce optimized loop code", and "performing a malicious code detection procedure on the optimized loop code". Teblyashkin is not concerned with code containing decryption loops and makes no mention of a decryption loop. As a result, claim 27 is not anticipated by Teblyashkin. Therefore Applicant respectfully requests that the Examiner reconsider the rejection and withdraw it.

Response to Rejections Under 35 USC § 103

In the Office Action, the Examiner rejected claims 3, 4, 6-19, 24-26, and 28 under 35 USC § 103(a) as being unpatentable over Christodorescu in view of U.S. Patent No. 5,826,013 ("Nachenberg"). This rejection is respectfully traversed.

Claim 6 recites a computer-implemented method for determining whether computer code contains malicious code, the method comprising:

identifying computer code suspected of currently containing malicious code, the computer code having a decryption loop and a body;

optimizing the decryption loop to produce optimized loop code;

performing a malicious code detection procedure on the optimized loop code;

optimizing the body to produce optimized body code;

subjecting the optimized body code to a malicious code detection protocol; and

responsive to the malicious code detection procedure detecting malicious code in the optimized loop code or the malicious code detection protocol detecting malicious code in the optimized body code, declaring a confirmation that the computer code contains malicious code.

(emphasis added)

As can be seen, claim 6 recites identifying computer code suspected of currently containing malicious code, where the computer code has a decryption loop and a body. The decryption loop is optimized to produce optimized loop code and a malicious code detection procedure is performed on the optimized loop code. A confirmation of malicious code is declared responsive to detecting malicious code in the optimized loop code or in optimized body code. By optimizing the decryption loop, the claimed invention beneficially removes possible obfuscations in the decryption loop so that a malicious code detection procedure can be more efficient and successful. The optimization can also result in faster emulation of the decryption loop to decrypt the body of the computer code. Support in the specification is found, for example, on page 4, line 16 to page 5, line 21 and on page 10, lines 21-27.

Claim 6 is not obvious in view of Christodorescu and Nachenberg. Christodorescu discloses converting a program to a "standardized version" that expresses the function of the program and using this standardized version for malware detection. Nachenberg describes a method for detecting a polymorphic virus using emulation, similar to the method described in the Background Art of the present specification. Nachenberg discloses emulating a decryption loop to decrypt a virus body so that the decrypted body can be compared to known viruses. Nachenberg discloses reducing the number of instructions to be emulated, but this reduction is not done through code optimization.

The references do not show "optimizing the decryption loop to produce optimized loop code," or "performing a malicious code detection procedure on the optimized loop code." The Examiner cites Nachenberg, col. 1, lines 63-67, and col. 2, lines 1-25, and Christodorescu, paragraph [0011], as disclosing each of these elements. However, the cited portions of Nachenberg merely describe examining sequences of instructions during emulation to determine if those sequences are likely part of a decryption loop (e.g., the sequences contain "boosters"). If it is determined that the sequences are not likely part of a decryption loop, emulation can be stopped. The cited portion of Christodorescu discloses creating a standardized version of a program and comparing the standardized version to standardized malicious code portions. Neither portion discloses optimizing a decryption loop and performing a malicious code detection procedure on the optimized code.

In the "Response to Arguments" section of the Office Action, the Examiner additionally cites Nachenberg, col. 6, line 54 to col. 7, line 8. This portion of Nachenberg describes the use of a static exculsion module 230 and a dynamic exclusion module 240 to "substantially reduce the number of file instructions that must be emulated." The Examiner states that this substantial reduction of file instructions can be interpreted as "code optimization." However, while code optimization may result in a reduction of instructions, a reduction of emulated instructions does not imply that code optimization has taken place, or that optimized code as been produced (as recited in claim 6).

The static exclusion module 230 in Nachenberg merely looks at "gross features of the executable image 100 that are inconsistent with various polymorphic viruses." (Nachenberg, col. 7, lines 9-20). The static exclusion module 230 then completely discards such files without performing emulation on them. While this "reduces the number of file instructions that must be emulated," as stated in Nachenberg it can hardly be considered "optimizing" to produce "optimized code" as required by claim 6.

The dynamic exclusion module 240 of Nachenberg accesses instruction/interrupt usage profiles 224 of known viruses during emulation to determine whether the currently emulated code may be part of a virus decryption loop (Nachenberg, col. 6, line 65 – col. 7, line 2). The dynamic exclusion module 240 "... determines when emulation has proceeded to a point where at least some code from the decrypted static virus body (160) may be scanned and substantially reduces the number of instructions emulated prior to scanning ..." (Nachenberg, col. 3, lines 48-52). The dynamic exclusion module 240 therefore reduces the number of emulated instructions by stopping emulation early, not by optimizing to produce optimized code.

Based on the above remarks, Applicant submits that for at least these reasons a person of ordinary skill in the art would not find the invention as defined in claim 6 or dependent claims 3, 4, 7-19, 24-26, and 28 to be obvious over the cited references. Therefore, Applicant respectfully requests that the Examiner reconsider the rejection and withdraw it.

Regarding dependent claims, claim 9 recites "wherein the step of optimizing the body comprises using at least one output from the group of steps consisting of optimizing the decryption loop and performing a malicious code detection procedure on the optimized loop code." The Examiner cites Christodorescu, paragraph [0011], and Nachenberg, col. 3, lines

35-53, as disclosing this element. However, the cited portion of Nachenberg merely discloses emulating a possible decryption loop and scanning the virus body after sufficient emulation. It discloses reducing instructions to be emulated through various exclusion modules. The cited portion of Christodorescu discloses creating a standardized version of a program and comparing the standardized version to standardized malicious code portions. Neither portion discloses using an output of optimizing a decryption loop for optimizing the body. Also, neither portion discloses using an output of performing a malicious code detection procedure on the optimized loop code for optimizing the body.

Claim 14 recites “performing a backward pass operation.” The Examiner cites paragraphs [0018] and [0021] of Christodorescu as disclosing this element. However, the cited portions do not mention a backward pass portion of an optimization. These portions discuss providing a functional expression of code and removing non-executing program portions.

Claim 28 recites “wherein the malicious code detection procedure comprises emulating the optimized loop code.” The Examiner cites paragraph [0027] of Christodorescu as disclosing this element. However, the cited portion does not mention emulation.

Applicant invites Examiner to contact Applicant’s representative at the number provided below if Examiner believes it will help expedite furtherance of this application.

Respectfully Submitted,
FREDERIC PERRIOT

Date: April 14, 2009

By: /Nikhil Iyengar/

Nikhil Iyengar, Reg. No. 60,910
Attorney for Applicant
Fenwick & West LLP
801 California Street
Mountain View, CA 94041
Tel.: (415) 875-2367
Fax: (650) 938-5200